LIBRARY OF THE
UNIVERSITY OF ILLINOIS
AT URBANA-CHAMPAIGN

510.84
I l6r
no. 308-315
cop. 2

The person charging this material is responsible for its return to the library from which it was withdrawn on or before the **Latest Date** stamped below.

Theft, mutilation, and underlining of books are reasons for disciplinary action and may result in dismissal from the University.

UNIVERSITY OF ILLINOIS LIBRARY AT URBANA-CHAMPAIGN

APR 26 1974

APR 9 Recd

L161—O-1096

Digitized by the Internet Archive
in 2013

http://archive.org/details/paxiipictureproc314boro

.G6r
314
.2

Report No. 314

COO-1018-1174

THE PAX II PICTURE PROCESSING SYSTEM AT THE UNIVERSITY OF ILLINOIS PROGRAMMING MANUAL

edited by

Richard T. Borovec

THE LIBRARY OF THE
OCT 2 1969
UNIVERSITY OF ILLINOIS
AT URBANA-CHAMPAIGN

March 18, 1969

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

COO-1018-1174

Report No. 314

THE PAX II PICTURE PROCESSING SYSTEM AT THE UNIVERSITY OF ILLINOIS PROGRAMMING MANUAL

edited by

Richard T. Borovec

March 18, 1969

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

The PAX II Picture Processing System, written at the University of Maryland, currently exists on the 360/75 at the Department of Computer Science at the University of Illinois.

This file note represents the Programming Manual for PAX. It consists primarily of Technical Report 68-67 of the University of Maryland Computer Science Center. The JCL necessary to use PAX, which resides as a load module on UIDCS3, precedes TR 68-67.

Any questions, comments, discovery of bugs, or suggestions for additions should be directed to the Illiac III Secretary in 297 Digital Computer Lab. Further copies of this manual may be obtained from the DCL Mail Room.

JCL AND DECK SETUP TO USE PAX II

```
/*ID                    (ACCOUNTING INFORMATION)
/*360                   (IF NECESSARY)
//        EXEC    FORTLKGO,REGION.GO=162K
//FORT.SYSIN DD *
```

PS=1266
DEPT=DCS
NAME=YOUR NAME

```
        FORTRAN IV MAIN PROGRAM
/*
//LKED.PAXLIB DD UNIT=2314,VOLUME=SER=UI[illegible],DISP=([illegible],PASS),
//                DSNAME=USER.P1142.ILLIAC3
//LKED.SYSIN DD *
   INCLUDE      PAXLIB(PAX)
/*
//GO.SYSIN DD *
```

USR1

SHR

P1266

```
        DATA DECK
/*
```

Technical Report 68-67 May 1968

The PAX II Picture Processing System

by

James W. Snively, Jr.*

and

Edgar B. Butt

* Now with the Sun Oil Company, Philadelphia, Pa.

The development of PAX II was supported in part by the National Institutes of Health under Contract PH-43-67-1099 and in part by the National Aeronautics and Space Administration under Grant NsG 398.

TABLE OF CONTENTS

1. FOREWORD 1

2. Description of PAX II 5

2.1 Planes 5

2.2 Windows 7

2.3 Stacks of planes 8

2.4 Shifting 9

2.5 Directions and direction lists 10

2.6 Data structure 11

2.7 Labeled Common 13

3. Subroutines 15

3.1 Key to subroutine parameters 15

3.2 Index of user subroutines 16

3.3 User subroutines classified by function . . 20

3.4 Index of system subroutines 21

3.5 List of assembly language subroutines . . . 22

3.6 Documentation of subroutines 23
(alphabetical order)

1. Foreword

Over the past fifteen years, a body of techniques for processing pictorial information by computer has been developed. Many of these techniques involve operations which are "uniform" (or "space-invariant") in the sense that they do not depend on position within the picture. For example, "blurring" a picture (the new value at each point is the average of the old values over a given neighborhood of the point), or "differentiating" it (e.g., the new value is the magnitude of the gradient of the old value), are uniform operations, since their definitions are independent of position.

In these examples, the operations could in principle be performed "in parallel", i.e., simultaneously at every point of a picture, since their effect at any point does not depend on their effects on other points. If a computer could perform many operations in parallel, it would be possible to process pictures very quickly.

Although conventional digital computers normally execute only a single arithmetical operation at a time, it is possible to achieve a certain degree of parallelism by taking advantage of the fact that most computers can perform basic logical and shifting operations simultaneously on each bit of a "word".

This implies a potential saving in picture processing time by a factor equal to the word length, typically 32 or 36.

Even if a picture contains many gray levels, one can regard it as a stack of binary pictures (each "bit plane" specifying one binary place of the gray level value), and perform arithmetical operations on the stack by applying appropriate sequences of logical operations to the individual bit planes; this still allows considerable savings over sequential processing.

An IBM 7090/94 program designated PAX, which performs operations on digital pictures using the approach just described, was written at the University of Illinois during 1963 [1-3]. [This program simulates the "pattern articulation unit" (PAU) of the ILLIAC III computer, which can perform operations in parallel on digitized pictures stored in two-dimensional registers.] A CSC 3600 version of PAX, called COMPAX, has been implemented at the Tata Institute of Fundamental Research, Bombay, India [4]. PAX has also been implemented at the National Bureau of Standards in IBM 7094 versions called STANDPAX and SPIFII (STANDPAX Imbedded in FORTRAN II).

The present report describes a revised and expanded version of PAX, designated PAX II, developed at the University of Maryland during 1967. The chief features of this version are:

(1.) PAX II is a collection of subroutines which may be called by a program written in FORTRAN (any version equivalent to FORTRAN IV). Hence the PAX II user has at his disposal the computational and testing features of FORTRAN.

(2.) Most of the PAX II subroutines are written in FORTRAN to facilitate implementation on various computers. At present, PAX II has been implemented on the IBM 7094, IBM 360 and Univac 1108 computers.

(3.) The pictures operated on by PAX II, except for several computer dependent restrictions, are arbitrary in size and number. Fast access external storage can be used, if available.

In the remainder of this report, PAX II will be referred to for brevity as PAX. It is assumed that the reader is familiar with FORTRAN.

References

[1] J. H. Stein, "User's manual for PAX, an IBM 7090 program to simulate the pattern articulation unit of ILLIAC III", Report No. 147, Digital Computer Laboratory, University of Illinois, July 1963.

[2] _____, "Program description of PAX, an IBM 7090 program to simulate the pattern articulation unit of ILLIAC III", Report No. 151, Digital Computer Laboratory, University of Illinois, September 1963.

[3] R. Narasimhan, "Labeling schemata and syntactic descriptions of pictures", Info. Control 7, July 1964, 151-179.

[4] _____, "Syntax-directed interpretation of classes of pictures", Comm. Assoc. Comp. Mach. 9, March 1966, 166-173.

2. Description of PAX

2.1 Planes

The basic data unit for PAX is the plane, a matrix whose elements are either zero or ones. An element, or point, in a plane is identified by its column number and row number. (Note that this coordinate notation differs from both Cartesian and matrix notations.) The dimensions of the planes are determined by parameters stored in a labeled common, subject to the restriction that the number of elements in a row must be a multiple (at least two) of the number of bits in a computer word. On the IBM 7094 and UNIVAC 1108, the number of elements in a row may be 72, 108, 144, etc. On the IBM 360 the number may be 64, 96, 128, etc. For different applications, plane size may be changed by altering the block data program, COMMON, which initializes labeled common. During execution of a given program, however, plane size remains fixed.

The number of planes available is also determined by parameters in labeled common, subject to restrictions of internal and external storage space available. In computers with fast access external storage, PAX has the capacity to store planes externally and to move planes or parts of planes to and from core storage as they are needed. The PAX user need not be concerned with details of when and how the planes are moved.

When a PAX user wants to use a plane, he chooses an integer variable, sets it to zero, and uses it as an output argument in a PAX subroutine. (Output arguments are those which begin with the letters IPO in the subroutine documentation in this manual.) PAX, when it detects the zero, selects an unused plane, stores a number identifying the plane in the integer variable, and performs the requested operation on the plane. Subsequent use of the same integer variable as an argument in a PAX subroutine will cause the same plane to be operated upon. A FORTRAN integer variable so used to identify a plane will be called a plane index. The following FORTRAN instructions will select a plane, set all points to zero, and then set the third point in the first row to one.

```
IP = 0
CALL CLEAR (IP, 0)
CALL WRITEC (IP, 3, 1)
```

If all planes are in use and a request for an unused plane is attempted, PAX provides a core dump and then terminates execution of the program. To help avoid this situation, planes that are in use but no longer needed should be released, that is, returned to unused status by calling subroutine RELEAS. Assume that plane IP is in use. The following instructions select a new plane, IPS, put on plane IPS the result of shifting plane IP one element to the right, combine the two planes on plane IP with a logical OR, and release plane IPS. The same

number of planes is in use before and after the instructions are executed.

```
.
.
.
.
.

IPS = 0
CALL SHIFT (IPS, IP, 1, 0)
CALL OR (IP, IP, IPS, 0)
CALL RELEAS (1, IPS)

.
.
.
.
.
```

The user must keep in mind that the operation of many PAX subroutines requires planes in addition to those he is using. Any subroutine using such "scratch planes" will release them when it completes its operation.

Some of the PAX subroutines have an argument called a context plane. In general, a context plane is a binary plane used as a mask to limit the operation of a subroutine to a subset of the points on a plane or stack of planes.

2.2 Windows

Some PAX routines operate on rectangular subsets of a plane known as windows. A window is specified by an array of four integers which represent respectively 1) the first column, 2) the first row, 3) the width, and 4) the height, of the window.

The following statements will define IW as a ten by ten window in the upper left-hand corner of a plane and then will cause that window of plane IPX to be printed.

```
DIMENSION IW(4)
DATA IW /1, 1, 10, 10/
.
.
.
CALL PRINT (1, IW, IPX)
.
.
.
```

2.3 Stacks of Planes

To represent a grayscale picture--a matrix whose elements are non-negative integers that may be greater than one--PAX uses an array or "stack" of planes. The value of a point in the jth column, kth row of a stack of planes IP(1),...,IP(NP) is

$$\sum_{i=1}^{NP} 2^{i-1} M_i$$

where M_i is the value (0 or 1) of the point in the jth column, kth row of plane IP(i). Subroutine WRITEZ may be used to set a point in a stack of planes to a given value. Subroutine READZ sets an integer variable to the value of a specified point in a stack of planes. Subroutines INPUT and OUTPUT transmit values between an integer array and a window of a stack of planes, one row at a time.

The following instructions establish a stack of three planes, set all points to zero, read values from the first ten columns of ten cards into a ten by ten window in the upper left corner of the stack, and then set JVALUE to the value of the point in the first column, fifth row of the stack.

```
      DIMENSION IP(3), IW(4), K(10)
      DATA IP/3*0/, IW/ 1,1,10,10/
      CALL CLEARS(IP, 0, 3)
      CALL INPUT (3, IW, IP, K, JGT)
10    READ (5,20) K
20    FORMAT (10I1)
      CALL INROW
      GO TO (10, 30), JGT
30    CALL READZ (JVALUE, 1, 5, 3, IP)
```

The user must insure that stacks of planes are large enough for the values they will contain. A program may operate on stacks of different sizes, since subroutines that operate on stacks require the size of the stack as an argument.

2.4 Shifting

Shifting operations in PAX require the user to specify two integer shift parameters which indicate the number of points the plane is to be shifted horizontally and vertically. Positive parameters indicate right and up shifts; negative parameters indicate left and down shifts. Zeros are shifted onto the edges of the plane opposite the direction of the shift.

The following instructions select a new plane, IP, set all points to one, shift the ones off all edges of the plane, and complement the plane. The result is a plane with ones only along its edges.

```
IP  = 0
CALL CLEAR (IP,1)
CALL SHIFT (IP, IP,2,2)
CALL SHIFT (IP,IP,-1,-1)
CALL EQUAL (IP,IP,1).
```

2.5 Directions and Direction Lists

Pairs of shift parameters are called directions. The identity direction is 0,0. A direction list is an array of shift parameters and has the following form: The first element of the array specifies how many directions follow in the array. Then successive pairs of elements of the array are the directions. Some subroutines require a direction list as an argument. Other subroutines require the location of a direction list, that is, an address constant point to the first element of a direction list. Integer Function LOC generates such address constants.

The following instructions define a direction list, IDL, containing five directions, and makes LDL its location.

```
DIMENSION IDL(11)
DATA IDL/5, 0,0, -1,0, 0,1 1,0, 0,-1/
LDL=LOC(IDL)
```

Direction lists are used to define neighborhood and connectedness. Let IDL be a direction list. The IDL-neighbors of a point are the points obtained by shifting the given point in the directions in the list. An IDL-connected string of points in a plane is a finite sequence of points $P_0, P_1, \ldots P_n$ such that all the points have the same value and P_i is an IDL-neighbor of P_{i-1} for $i = 1, 2, \ldots, n$. A point P is IDL-connected to a point P_0 if there is an IDL-connected string $P_0, \ldots, P_n$ such that $P = P_n$.

2.6 Data Structure

The points in one row of a plane are represented by successive bits of consecutive computer words. Successive rows are stored one after the other. Planes may be divided horizontally into blocks containing equal numbers of rows. Except in the case described below, one block per plane should be specified. Fast access external storage may be used to permit very large planes or to increase the number of planes available. When external storage is used, blocks are the data elements that are moved between internal and external storage. If internal storage is sufficient for only a few planes, then each plane must be divided into several blocks so that blocks of many planes may be in internal storage simultaneously.

Blocks of planes are stored internally in buffers. Subroutine LOCATE controls the buffer assignments and calls Sub-

routine DISK, when necessary, to move blocks between buffers and external storage. Subroutine DISK controls external storage allocation.

When a PAX subroutine needs a block of a particular plane, it calls Subroutine LOCATE, which finds the block. If the block is not in a buffer, an available buffer is found or a buffer is made available by moving its contents to external storage, and then the specified block is moved into the buffer. The buffer is marked so that its contents are ineligible to be moved to external storage, and the address of the buffer is returned to the calling subroutine. When the subroutine is finished with the block it calls LOCATE which marks the buffer so that its contents are eligible for external storage.

To minimize movement of blocks between internal and external storage, Subroutine LOCATE maintains an activity queue of buffers. Unused buffers are kept at the head of the queue. When a buffer is active it is moved to the end of the queue. If a buffer is needed and none are unused, the contents of the buffer at the head of the queue--the least active buffer--are moved to external storage.

Subroutine LOCATE also supervises the status of the planes. If a subroutine passes a plane index of zero to

LOCATE, it selects an unusued plane, stores its number in the index and marks the plane as being in use. Planes are returned to unusued status by Subroutine RELEAS which is an entry in Subroutine LOCATE.

2.7 Labeled Common

The following blocks of labeled common are used by PAX. Some of these blocks are initialized by a block data program. Others are initialized on the first call to Subroutine LOCATE. Some of the blocks are of interest to the user because they contain parameters referring to plane size. By using these parameters, a program may be written that will operate properly if the plane size is altered.

A. Machine dependent blocks

```
/MWX/ NEW, NAW
/PRX/ NWL, NCH, FL1(3), FL2(3)
```

NEW - Number of bits in a word (integer variable)
NAW - Number of machine addresses per word
NWL - Number of words (packed with A-format characters) per print line.
NCH - Number of characters (excluding carriage control) per print line.
FL1 - Format for printing one line of packed A-format characters with single spacing.
FL2 - Format for printing one line of packed A-format characters with overstrike.

B. Blocks specifying plane size and number.

```
/LSZ/  LPB, LBA, LBR, LRE, LCE, LRA
/PLX/  NPLN, NBUF
/WEP/  IW(4)
```

LPB - Length of plane in blocks.
LBA - Length of block in addresses.
LBR - Length of block in rows.
LRE - Length of row in elements (bits).
LCE - Length of column in elements (bits).
LRA - Length of row in addresses.
NPLN - Number of planes.
NBUF - Number of buffers (each holds one block of a plane).
IW - Window for the entire plane.

C. Blocks used for storage. Dimensions of arrays are given in terms of previously mentioned parameters.

```
/PIX/  PLANES(NPLN)
/BUX/  BLOCK (5,NBUF)
/BUF/  BUFFER(LBA/NAW,NBUF)
/ROW/  IR(MAXO(LRE,NCH) )
```

PLANES - Array indicating status of each plane.
BLOCK - Array containing five control words for each buffer.
BUFFER - Buffers for storing blocks of planes.
IR - Array which may be used on a temporary basis to store values from one row of a plane.

D. Characters used by Subroutine PRINT

```
/LTX/LT(32)
```

LT - Array containing characters used by Subroutine PRINT.

3. Subroutines

3.1 Key to subroutine parameters

The conventions described below are followed in the subroutine documentation in this manual. Programmers using PAX are urged to follow these conventions in their own programs and documentation. Stars may be replaced by any valid character.

*GT	Computed go to parameter
IDL***	Direction list
INV***	Inverse switch associated with plane IP***
IP****	Plane index or array of plane indices
IPC***	Index of a context plane
IPI***	Index or array of indices of input planes
IPO***	Index or array of indices of output planes
IW****	Window of a plane
J*****	Integer argument which may be altered by a subroutine
KX	Column of a plane
KY	Row of a plane
LDL	Address constant or array of address constants pointing to direction lists
NP****	Number of plane indices in array IP****
NX	Horizontal shift parameter
NY	Vertical shift parameter
R*****	Real argument which may be altered by a subroutine

3.2 Index of user subroutines

Subroutines preceded by '=' are functions. All other subroutines are referenced by a 'CALL' statement. Cross-references are given on subroutines whose documentation is not in alphabetical order.

The first time a plane index is used, it must be used as one of the arguments beginning with the letters IPO.

ADD1TP(IP, IPS, NPS)

ADDSUB(IPO, NPO, IPI1, NPI1, IPI2, NPI2, IADSB)

AND(IPO, IPI1, IPI2, INVI2)

ANDS(IPO, IPI1, IPI2, INVI2, NP)

APPLY(IPO, NPO, IPOS, IPI, LDL)

AREA(IP, INV, IPC, JAREA)

BOOFUN(IPO, IPI, IPC, FUN)

BORDER(IP, JGT)

CARDS(NPO, IW, IPO)

CIRCLE(IP, KX, KY, KRAD)

CLEAR(IPO, INVO)

CLEARS(IPO, INVO, NPO)

COMPAN(LDL, IP, INV, JARRAY, JN)

CONNEC(LDL, IPO, IPI, IPC, JCOUNT)

=COORD(KX, KY) INTEGER COORD

CPROP(IPO, IPI, IPC, KRAD)

DIST(IPO, NPO, IPI, INVI, LDL, NDL, MAX)

DLTOIR(LDL, JX, JY, JGT, JID)

EDGES(IPO, IPI, LDL)

EQUAL(IPO, IPI, INVI)

EQUALS(IPO, IPI, INVI, NP)

EXCHAR

EXOR(IPO, IPI1, IPI2, INVI2) See AND

EXORS(IPO, IPI1, IPI2, INVI2, NP) See AND

EXTRM(JVALUE, MINMAX, IP, NP, IPC)

EXTRMP(JVALUE, MINMAX, IP, NP, IPC, IPO)

FRAME(IP, NSIZE)

GRID(IPO, NSIZE)

INETIA(NP, IP, IW, JX, JY, RECC, RANGLE)

INPUT(NPO, IW, IPO, NROW, JGT)

INROW

INSERT(IPO, NPO, IPI, NVALUE)

LETTER

LISTXY(IP, IW, JARRAY, J)

=LOC(IDL)

MARK(LDL, IPO, IPI, IPC)

MAXIMA(LDL, IPO, IPI, NPI)

MINIMA(LDL, IPO, IPI, NPI)

NEIGHB(IP, NP, KX, KY, IDL, JARRAY)

NULL(IP, JGT)

OR(IPO, IPI1, IPI2, INVI2) See AND

ORS(IPO, IPI1, IPI2, INVI2, NP) See AND

OUTPUT(NP, IW, IP, JROW, JGT)

OUTROW

=POINT(LOCATN, N)

PRINT(NP, IW, IR)

PROP(LDL, NDL, IPO, IPI, IPC, NPROPS)

PUNCH(NP, IW, IP)

=RANDNO(NR)

RANPIC(IPO, IW, PROB, NVALUE, NR)

READZ(JVALUE, KX, KY, NPI, IPI)

RELEAS(NP, IP)

RPROP(IPO, IPI, IPC, NX, NY)

SHIFT(IPO, IPI, NX, NY)

SHIFTS(IPO, IPI, NX, NY, NP)

SLICE(IPO, IPI, NPI, KREL, NUM)

SNAP

SPRINT(NP, IW, IP, NSP) See PRINT

SUMONE(IP, NP, IPS, IPI, IADSB)

TMARK(LDL, IPO, IPI, IPC, KREL, NUM)

TSHIFT(IP, NX, NY, JGT)

UNCOOR(KP, JX, JY)

WINDOW(IPO, IW)

WRITEC(IP, KX, KY)

WRITEZ(NVALUE, KX, KY, NP, IP)

ADDITIONS

COMBIN(IPO, IPIØ, IPI1, NP, IPM)

COMPAR(IPO, IPI1, IPI2, NPI, KREL)

ENDRP — See READP

ENDWP — See WRITEP

GLHS (NP, IW, IP, JHIST, NHIST)

GMARK (IPO, IPI, NP, IPC, LDL, MINMAX)

GREDGE (IPO, IPI, NPI, LDL)

LABEL (IPO, NPO, L, NL, KX, KY, NVALUE, MAG)

= LDL(K)

NAND (IPO, IPI1, IPI2) — See AND

NANDS (IPO, IPI1, IPI2, NP) — See AND

NORS (IPO, IPI1, IPI2, NP) — See AND

POLYGN (IPO, NPO, NVALUE, NVERT, COORDS)

PRINTS (NP, IW, IP)

PTHS (IHIST, NHIST, I)

READP (IPO, NPO, KX, KY, JVALUE)

ROTATE(IPO, NP, IWO, IPI, CXI, CYI, THETA, ALPHA, ISW)

RP

WP — See WRITEP

WRITEP(IPO, NPO, KX, KY, NVALUE)

3.3 User subroutines classified by function

This classification does not include all user routines. It is intended as a guide for those not familiar with PAX.

A. Input to planes
WRITEX, WRITEZ, INPUT, CARDS

B. Output from planes
READZ, NEIGHB, LISTXY, OUTPUT, PRINT, PUNCH

C. Figure creation
CIRCLE, FRAME, GRID, WINDOW, INSERT

D. Logical operations
CLEAR, EQUAL, AND, EXOR, OR, BOOFUN

E. Arithmetical operations on stacks of planes
ADD1TP, SUMONE, ADDSUB, MAXIMA, MINIMA, SLICE

F. Testing and measuring operations
NULL, EXTRM, AREA, TSHIFT, BORDER, INETIA

G. Shifting and iterated shifting operations
SHIFT, MARK, PROP, CPROP, RPROP

H. Operations relating to connectedness
CONNEC, COMPAN, EDGES

3.4 Index of System Subroutines

The following subroutines, although available to the user, are primarily for use by other PAX subroutines. Subroutines preceded by '=' are functions. All other subroutines are referenced by a 'CALL' statement. Cross references are given on subroutines whose documentation is not in alphabetical order.

DISK(BUFFER)

FP(LWORD, BIT, ARRAY, N, POS)

=KGB(LOCATN, N)

=KGC(KV, K)

KPB(I, LOCATN, N)

KPC(KAR, KV, N)

LOCATE(NP, IP, IB, LOCATN)

OP(LOCATN) See SETOP

PF(LWORD, BIT, ARRAY, N, POS)

RSEEK(NPB) See DISK

SETOP(OPCODE, LENGTH)

SETSH(COUNT, LENGTH)

SH(LOCIN, LOCOUT) See SETSH

WSEEK(NPB) See DISK

ADDITIONS

= KGBS(LL, N, NPX)

3.5 List of assembly language subroutines

COORD

FP

KGB

KGC

KPB

KPC

LOC

PF

POINT

RANDNO

SETOP, OP

SETSH, SH

SNAP

UNCOOR

ADDITIONS

KGBS

3.6 Documentation of Subroutines

Subroutine ADD1TP

Purpose:

To add the values in a plane to the values in a stack of planes.

Usage:

CALL ADD1TP(IP, IPS, NPS)

Parameters:

IP - Index of plane to be added
IPS - Array of indices of planes in the stack to which plane IP is to be added
NPS - Number of planes in stack IPS

Execution:

Plane IP is added to the stack IPS(1),...; IPS(NPS). Overflow from the stack is lost and is not detected.

Subroutine ADDSUB

Purpose:
To add or subtract two stacks of planes.

Usage:
CALL ADDSUB (IPO, NPO, IPI1, NPI1, IPI2, NPI2, IADSB)

Parameters:

IPO - Output stack
NPO - Number of planes in stack IPO
IPI1 - First input stack
NPI1 - Number of planes in stack IPI1
IPI2 - Second input stack
NPI2 - Number of planes in stack IPI2
IADSB - Integer set to 1 for addition or 2 for subtraction

Execution:
Stack IPI2 is added to (IADSB=1) or subtracted from (IADSB=2) stack IPI1 and the result is put in stack IPO. The stacks of planes need not be distinct. Some restriction is put on the relative sizes of the stacks. NPI2 must not exceed NPI1, and NPI1 must not exceed NPO.

If there is addition overflow or if subtraction would result in a negative value an error message is printed and execution is terminated.

Subroutine ADJACE

Purpose:

To determine all the positive values in a stack which are adjacent to a given value.

Usage:

CALL ADJACE (IP, NP, NVALUE, LDL, JARRAY, J)

Parameters:

IP	- Array of plane indices
NP	- Number of planes in stack IP
NVALUE	- Value indicating points on stack IP whose adjacent values will be determined
LDL	- Address constant pointing to direction list IDL which defines adjacency
JARRAY	- Array in which adjacent values will be stored
J	- Number of values stored in JARRAY by the subroutines

Execution:

Each positive integer which is the value of a point which is an IDL-neighbor of a point with value NVALUE is stored in JARRAY. J is set to the number of such integers. Care must be taken to dimension JARRAY sufficiently large.

Programmer:

Edgar Butt

Subroutines AND, ANDS, EXOR, EXORS, OR, ORS

Purpose:

To set each point of a plane (stack of planes) to a logical combination of the corresponding points on two planes (stacks of planes).

Usage:

CALL XXX(IPO, IPI1, IPI2, INVI2)
CALL XXXS(IPO, IPI1, IPI2, INVI2, NP)
XXX is AND, EXOR, or OR

Parameters:

IPO	-	Array of indices of output planes
IPI1	-	Array of indices of input planes
IPI2	-	Array of indices of input planes
INVI2	-	Inverse switch associated with IPI2
NP	-	Number of indices in IPO, IPI1, and IPI2

Execution:

Points in plane IPO(I) are set according to the values of corresponding points in planes IPI1(I) and IPI2(I) for I = 1, ..., NP according to the following table. For Subroutines, AND, EXOR, and OR, NP is assumed to be 1.

IPI1(I)	IPI2(I)	IPO(I)					
		AND, ANDS		EXOR, EXORS		OR, ORS	
		INV2=0	INV2≠0	INV2=0	INV2≠0	INV2=0	INV2≠
0	0	0	0	0	1	0	1
0	1	0	0	1	0	1	0
1	0	0	1	1	0	1	1
1	1	1	0	0	1	1	1

Subroutine APPLY

Purpose:

To determine the difference of the sums of the values of the points in two specified neighborhoods of each point of a plane. This is equivalent to cross-correlating of template, having values 0, +1, or -1 at every point, with the plane.

USAGE:

CALL APPLY (IPO, NPO, IPOS, IPI, LDL)

Parameters:

IPO	- Stack of planes to receive absolute value of the differences
NPO	- Number of planes in IPO
IPOS	- Plane containing the signs associated with the values in IPO
IPI	- Index of input plane
LDL	- Array containing the locations of two direction lists

Execution:

The absolute value stack and the sign plane are cleared. Plane IPI is shifted the opposite of each direction in the list pointed to by LDL(1) and added to the absolute value stack. Then plane IPI is shifted the opposite of each direction in the list pointed to by LDL(2) and algebraically subtracted from stack IPO and its associated sign plane IPOS. When the subroutine is finished, points on plane IPOS have the value 1 if the corresponding value on stack IPO is negative.

The user must see that stack IPO has enough planes to contain any possible result. Overflow is not detected.

Subroutine AREA

Purpose:

To find the number of ones or zeros in a plane which correspond to the ones in a context plane.

Usage:

CALL AREA (IP, INV, IPC, JAREA)

Parameters:

IP - Index of the plane whose area is being determined
INV - Inverse switch associated with plane IP
IPC - Index of the context plane (if not zero)
JAREA - Variable which is set to the area

Execution:

If IPC=0, JAREA is set to the number of ones (if INV=0) or zeros (if INV≠0) in plane IP.

If IPC≠0, the counting operation is restricted to points of plane IP which correspond to points of plane IPC which are one.

Subroutine BOOFUN

Purpose:

To set each point of a plane to a Boolean function of the neighbors of the corresponding point in a specified plane.

Usage:

CALL BOOFUN (IPO, IPI, IPC, FUN)

Parameters:

IPO - Index of output plane
IPI - Index of input plane
IPC - Index of context plane
FUN - Array containing numeric and special characters which specify the Boolean function. Valid special characters are: BLANK / ( ) - , + * $

Boolean Function:

Elements of the function are of the form (NX,NY) or /(NX,NY) where NX and NY are strings of digits, possibly preceded by a minus sign. (NX,NY) represents the result of shifting plane IPI with the parameters specified by NX and NY. /(NX,NY) is the complement of (NX,NY). Products (logical and) of elements are represented by elements separated by asterisks or merely by a string of elements. (0,0)*(1,1)*/(-1,-1) and (0,0)(1,1)/(-1,-1) are equivalent products. An entire product may be complemented by a following slash as in (0,1)*(-1,-1)/. Finally, products may be combined with the logical or operation by means of the plus sign.

Blanks are ignored. The last character must be a dollar sign.

Examples of Boolean Functions:

```
(0,1)*(0,-1)*(1,0)*(-1,0)$

(0,1)(0,-1)(1,0)(-1,0)/ $

/(1,0) + /(2,0) + (3,0) + (4,0) $

(0,1)(0,-1)/ + (1,0)(-1,0) $

(-1,-1)+(0,-1)+(1,-1) $
```

Execution:

Plane IPO is set to the result of applying the Boolean function to plane IPI. Then if IPC is not 0, plane IPC is anded onto plane IPO.

FUN may be a Hollerith constant, an array that has been initialized by a Hollerith field in a data statement or an array into which the function has been read with A or H format.

Subroutine BORDER

Purpose:
To determine whether or not any points on the border of a plane have the value 1.

Usage:
CALL BORDER (IP, JGT)

Parameters:
IP - Index of plane being tested
JGT - Variable set by the subroutine

Execution:
JGT is set to 2 if plane IP has a point in its first row or column or last row or column with value 1. Otherwise JGT is set to 1.

Subroutine CARDS

Purpose:

To input values from cards to a window of a stack of planes.

Usage:

CALL CARDS (NPO, IW, IPO)

Parameters:

NPO - Number of planes in the stack (not more than 5)
IW - Window
IPO - Array of plane indices

Execution:

This subroutine requires data cards with characters for the window of the stack of planes punched on successive columns. Characters for each row of the window must begin in the first card column, and, if necessary, continue on successive cards. The card characters 0, 1, ..., 9, A,B, ...,V correspond to picture values 0, 1,...,31, ,Picture values which exceed the capacity of the stack are changed to zero. Card characters other than those mentioned will produce a picture value of zero.

If IPO is 0 (for I=1 or 2 or ...or NPO), then plane IP (I) is cleared before the cards are input.

Subroutine CIRCLE

Purpose:
To set the points on a circle to one.

Usage:
CALL CIRCLE (IP, KX, KY, KRAD)

Parameters:
IP - Plane index
KX - Column in which the center lies
KY - Row in which the center lies
KRAD - Radius of the circle

Execution:
A point in plane IP is set to one if the distance from the point to the center of the circle is between KRAD-1/2 and KRAD + 1/2. All other points in the plane are unaltered. The circle need not lie entirely on the plane.

Subroutine CLEAR, CLEARS

Purpose:

To set all of the points of a plane or a stack of planes to zero or to one.

Usage:

```
CALL CLEAR (IPO, INV)
CALL CLEARS (IPO, INV, NPO)
```

Parameters:

IPO - Array of plane indices
INV - Inverse switch
NPO - Number of indices in IP

Execution:

All points of plane IPO(I) are set to 0 if INV=0 or to 1 if INV $\neq$ 0 for I = 1,..., NPO. For Subroutine CLEAR, NPO is assumed to be one.

Subroutine COMBIN

Purpose:

To create a picture by combining specified parts of two pictures.

Usage:

CALL COMBIN (IPO, IPI0, IPI1, NP, IPM)

Parameters:

IPO - Array of indices of planes in output stack
IPI0 - Array of indices of planes in first input stack
IPI1 - Array of indices of planes in second input stack
NP - Number of planes in each of the stacks IPO, IPI0 and IPI1
IPM - Index of mask plane

Execution:

If the value of a point on plane IPM is 0 or 1, then the corresponding point on stack IPO is set to the value of the corresponding point on stack IPI0 or IPI1 respectively.

Programmer:

Edgar Butt

Subroutine COMPAN

Purpose:

To perform an analysis of the connected components in a plane.

Usage:

CALL COMPAN (LDL, IP, INV, JARRAY, J)

Parameters:

LDL - Address constant pointing to IDL, a direction list which determines connectedness

IPL - Index of plane in which connectedness is being analyzed

INV - If INV is zero, plane IP will be analyzed; otherwise the complement of plane IP will be analyzed

JARRAY - Integer array to receive the areas of the connected components. JARRAY must be dimensioned no smaller than the initial value of J

J - Variable set by the user to indicate the maximum number of components to record. J is reset by the subroutine to indicate the number of components found

Execution:

The areas of the first J connected components are put in JARRAY. The search for connected components ends when all have been found or when the (J+1)-st component is found. On exit, J is reset to the number of components found.

Subroutine COMPAR

Purpose:

To make a comparison of the values of corresponding points of two stacks of planes.

Usage:

CALL COMPAR (IPO, IPI1, IPI2, NPI, KREL)

Parameters:

IPO - Index of plane which will contain the result of the comparison
IPI1 - Array of indices of planes in first input stack
IPI2 - Array of indices of planes in second input stack
NPI - Number of planes in stacks IPI1 and IPI2
KREL - Integer specifying a relation. Possible values and their corresponding relations are:

1 - less than
2 - less than or equal to
3 - equal to
4 - greater than or equal to
5 - greater than

Execution:

Each point on plane IPO is set to 1 if the values of corresponding points on stacks IPI1 and IPI2 satisfy relation KREL.

Programmer:

Edgar Butt

Subroutine CONNEC

Purpose:

To determine the points in a plane which are connected (according to some direction list) to any one of a specified set of points. Also, to determine the minimum number of calls to Subroutine MARK (using the same direction list) needed to expand the specified set to include all the connected points.

Usage:

CALL CONNEC(LDL, IPO, IPI1, IPI2, JCOUNT)

Parameters:

LDL	- Address constant pointing to a direction list, IDL, which determines connectedness
IPO	- Index of output plane
IPI1	- Index of plane in which connectedness is being examined
IPI2	- Index of plane which specifies a set of points
JCOUNT	- Variable to be set to the number of calls to subroutine MARK needed to expand the set of points specified by plane IPI2 to include all connected points

Execution:

A scratch plane, IPS, is initially set to the logical and of planes IPI1 and IPI2. Then Subroutine MARK is applied to plane IPS with direction list IDL and the plane IPI1 is anded onto the result. This process is repeated until it adds no new 1's to plane IPS. Then plane IPS is copied on plane IPO. JCOUNT is set to the number of calls to Subroutine MARK which added 1's to the output image.

This Subroutine operates more efficiently if the direction list does not contain the direction (0,0).

Integer Function COORD

Purpose:

To pack two positive integers into a single integer variable.

Usage:

```
INTEGER COORD
K = COORD(I, J)
```

Parameters:

I, J - Positive integers with not more than half the number of significant bits in an integer variable

Execution:

I and J are packed into the variable K. Subroutine UNCOOR performs the unpacking.

Subroutine CPROP

Purpose:

To generate a plane which has a circular disk of 1's (0's) with a given radius centered at each point corresponding to every 1 (0) on an input plane.

Usage:

CALL CPROP (IPO, IPI, IPC, KRAD)

Parameters:

IPO - Index of output plane
IPI - Index of input plane
IPC - Index of context plane
KRAD - Radius of propagation (positive or negative)

Execution:

If KRAD is not negative, each point in plane IPO is set to 1 if it is less than KRAD+½ units from a point corresponding to a 1 in plane IPI. If KRAD is negative, each point in plane IPO is set to 0 if it is less than |KRAD|+½ units from a point corresponding to a 0 in plane IPI. If IPC is not 0, then plane IPO may differ from plane IPI only on points corresponding to a 1 in plane IPC.

A unit is the distance between adjacent picture elements on the same row or column.

Subroutines DISK, WSEEK, RSEEK

Purpose:

To move blocks of planes between internal buffers and external storage.

Usage:

```
CALL  WSEEK (NPB)
CALL  RSEEK (NPB)
CALL  DISK (BUFFER)
```

Parameters:

NPB - Array of two words containing plane number and block number

BUFFER - Internal array which contains or will receive the block to be moved

Execution:

When WSEEK is called it prepares the external storage device to write block NPB(2) of plane NPB(1). RSEEK performs a similar task for reading.

When DISK is called it moves the block stored in BUFFER to external storage or moves a block from external storage to BUFFER according to the most recent call to WSEEK or RSEEK. The number of words in a block is determined by parameters in labelled common.

Subroutine DIST

Purpose:

To set the value of each point in a stack of planes to the distance from the point to the closest of a specified set of points. Distance is determined by an array of direction lists.

Usage:

CALL DIST (IPO, NPO, IPI, INVI, LDL, NDL, MAX)

Parameters:

- IPO - Array of plane indices
- NPO - Number of planes in stack IPO
- IPI - Index of plane which specifies set of points from which distance is measured
- INVI - Determines whether distance is measured from points in plane IPI with value 1 (IPI=0) or 0 (IPI≠0)
- LDL - Array of address constants pointing to direction lists used to determine distances
- NDL - Number of address constants in LDL
- MAX - Maximum distance to appear on stack IPO. Points with distance greater than MAX will have value MAX

Execution:

Stack IPO is cleared and the complement of plane IPI is added to the stack. Subroutine MARK is applied to plane IPI and the complement of the result is added to the stack. Subroutine MARK is applied repeatedly to the result of the previous call to MARK and the complements of the results are added to the stack. The NDL direction lists pointed to by elements of LDL are used cyclically in calls to MARK. The process is terminated when some points have achieved the value MAX or when overflow from the stack would occur.

Subroutine DLTOIR

Purpose:

To facilitate an instruction loop using successive shift parameters from a direction list.

Usage:

```
     JID=0
10   CALL DLTOIR(LDL,JX,JY,JGT,JID)
     GO TO (20,30), JGT
20   .
     .(Instructions using shift parameters JX and JY)
     .
     GO TO 10
30   .
     .
     .
```

Parameters:

LDL - Address constant pointing to a direction list IDL
JX - Horizontal shift parameter
JY - Vertical shift parameter
JGT - Computed go to parameter
JID - Variable set to indicate which pair of shift parameters has been returned

Execution:

IDL(1) is the number of pairs of shift parameters which follow in the array. If JID is less than IDL(1), JID is increased by one, JX and JY are set to the JIDth pair of shift parameters, and JGT is set to one. If JID is not less than IDL(1), JID is set to 0 and JGT is set to 2.

Subroutine EDGES

Purpose:

To find the edges, defined by a direction list, of all the objects in a plane.

Usage:

CALL EDGES(IPO, IPI, LDL)

Parameters:

IPO - Index of output plane
IPI - Index of plane containing objects being examined
LDL - Address constant pointing to direction list IDL which defines edges

Execution:

Each point of plane IPO is set to 1 if the corresponding point in plane IPI has the value 1 and is an IDL-neighbor of a point with value 0.

Subroutine EQUAL, EQUALS — complement !

Purpose:

To set a plane (stack of planes) equal to another plane (stack of planes) or its inverse.

Usage:

```
CALL EQUAL(IPO, IPI, INV)
CALL EQUALS (IPO, IPI, INV, NP)
```

Parameters:

IPO - Array of indices of output planes
IPI - Array of indices of input planes
INV - Inverse switch associated with IPI
NP - Number of indices in IPO and IPI

Execution:

If INV = 0, plane IPI(I) is copied in plane IPO(I) for I = 1, ..., NP. If INV≠0, the inverse of plane IPI(I) is put in plane IPO(I). For subroutine EQUAL, NP is assumed to be 1.

Subroutine EXCHAR

Purpose:

To exchange the print characters currently being used by Subroutine PRINT and the characters stored within this subroutine. Initially, characters whose blackness corresponds to the gray levels they represent are stored in this subroutine.

Usage:

CALL EXCHAR

Execution:

The contents of the arrays LT(32) (in the labeled common block/LTX/) and ALT(32) (in this subroutine) are exchanged.

See subroutine PRINT for a discussion of print characters.

Subroutines EXTRM, EXTRMP

Purpose:

To determine the extreme values (either minimum or maximum) of a stack of planes.

Usage:

```
CALL EXTRM (JVALUE, MINMAX, IP, NP, IPC)
CALL EXTRMP(JVALUE, MINMAX, IP, NP, IPC, IPO)
```

Parameters:

JVALUE	-	Integer variable to be set to minimum or maximum value
MINMAX	-	Integer indicating whether the minimum (MINMAX=1) or maximum (MINMAX=2) value will be found
IP	-	Array of indices of planes in the stack to be examined
NP	-	Number of planes in stack IP
IPO	-	Index of output plane (used by Subroutine EXTRMP only)

Execution:

JVALUE is set to the minimum or maximum value appearing on stack IP. Subroutine EXTRMP sets each point of plane IPO to 1 if the corresponding point on stack IP has the extreme value JVALUE.

Subroutine FP

Purpose:

To facilitate bit-plane encoding. Specifically, to set successive bits, beginning with a specified bit, to the values of the bits in a specified position in successive words of an array.

Usage:

```
CALL FP (LWORD, BIT, ARRAY, N, POS)
```

Parameters:

LWORD - Address constant pointing to the word containing the first bit that will have its value set
BIT - Number, counting from 1 on the left, of the first bit in the word pointed to by LWORD that will have its value set
ARRAY - Array of N words in which bits will be examined.
N - Number of bits to be set
POS - Word containing a single 1-bit which specifies the bit to be examined in each word of the array

Execution:

N consecutive bits beginning with the one specified by LWORD and BIT are set to the values of the bits in the position specified by POS in the words ARRAY(1),..., ARRAY(N).

Subroutine FRAME

Purpose:

To put a frame of ones on a plane.

Usage:

```
CALL FRAME(IP, NSIZE)
```

Parameters:

IP - Plane index
NSIZE - Size of frame in picture elements

Execution:

Each point on plane IP which is not more than NSIZE elements from the edge of plane IP is set to one. Other points on the plane are not altered.

Subroutine GLHS

Purpose:

To determine the frequencies with which values occur in a window of a stack of planes.

Usage:

```
CALL GLHS (NP, IW, IP, JHIST, NHIST)
```

Parameters:

NP	- Number of planes in stack IP
IW	- Window of stack IP to be examined
IP	- Array of plane indices
JHIST	- Array that will receive the frequencies with which values occur
NHIST	- Dimension of JHIST

Execution:

For I = 1,..., NHIST - 1, JHIST(I) is set to the number of points in window IW of stack IP which have the value I-1. JHIST(NHIST) is set to the number of points with value NHIST-1 or greater.

Programmer:

Li-Chen Wang

Subroutine GMARK

Purpose:

To determine the minimum or maximum value occurring in a neighborhood of each point in a stack of planes.

Usage:

CALL GMARK (IPO, IPI, NP, IPC, LDL, MINMAX)

Parameters:

IPO	- Array of indices of planes in the output stack
IPI	- Array of indices of planes in the input stack
NP	- Number of planes in stacks IPO and IPI
IPC	- Context plane index
LDL	- Address constant pointing to direction list IDL
MINMAX	- Integer set to 1 for minimum or 2 for maximum.

Execution:

Each point on stack IPO is set to the minimum (MINMAX = 1) or maximum (MINMAX = 2) of the values of the IDL-neighbors of the corresponding point on stack IPI. If IPC is not 0, then for each point on plane IPC with the value 0, the corresponding point on stack IPO will be set to the value of the corresponding point on stack IPI rather than the maximum or maximum previously described. Stacks IPO and IPI must be distinct.

Programmer:

Edgar Butt

Subroutine GREDGE

Purpose:

To determine the points on a stack of planes which differ in value from at least one of their neighbors.

Usage:

CALL GREDGE (IPO, IPI, NPI, LDL)

Parameters:

IPO - Output plane index
IPI - Array of indices of planes in the input stack
NPI - Number of planes in stack IPI
LDL - Address constant pointing to direction list IDL

Execution:

Each point of plane IPO is set to 1 if the corresponding point on stack IPI differs in value from any of its IDL-neighbors.

Programmer:

Edgar Butt

Subroutine GRID

Purpose:

To generate a square grid of arbitrary size on a plane.

Usage:

```
CALL GRID(IPO, NSIZE)
```

Parameters:

IPO - Plane index
NSIZE - Size of grid

Execution:

Plane IPO is cleared, and each point whose row number or column number is a multiple of NSIZE is set to 1.

Subroutine INETIA

Purpose:

To determine the centroid and principal axis of the contents of a window of a stack of planes. Also to determine the eccentricity of the ellipse whose major and minor axes are equal in magnitude to the maximum and minimum moments of inertia.

Usage:

CALL INETIA (NP, IP, IW, JX, JY, RECC, RANGLE)

Parameters:

- NP - Number of planes in the stack
- IP - Indices of planes in the stack
- IW - Window being examined
- JX, JY - Column and row of centroid
- RECC - Eccentricity described above
- RANGLE - Angle in radians ($-\pi \leqq 0 \leqq \pi$) from a horizontal line to the principal axis

Execution:

Values in the window are retrieved a row at a time using INPUT and INROW. The above parameters are calculated using well known formulas.

Subroutines INPUT, INROW

Purpose:

To put values in successive rows of a window of a stack of planes.

Usage:

```
      CALL INPUT(NPO, IW, IPO, NARRAY, JGT)
10    .
      . (Instructions to put the values for one row of the
      . window into NARRAY)
      .
      CALL INROW
      GO TO (10,20), JGT
20    .
      .
      .
```

Parameters:

NPO	-	Number of indices in IPO
IW	-	Window
IPO	-	Array of plane indices
NARRAY	-	Integer array large enough to hold values for one row of the window
JGT	-	Computed go to parameter set by the subroutine

Execution:

INPUT and INROW are entries to the same subroutine. When INPUT is called, the subroutine is initialized and JGT is set to 1. When INROW is called, the values for one row of the window which the user has put in NARRAY are put in the stack of planes. On the call to INROW for the last row of the window, JGT is set to 2.

If IPO(I) is 0 (for I=1 or 2 or ... or NPO), plane IPO(I) is cleared.

INSERT

Subroutine INSERT

Purpose:

To put an object on a binary plane into a stack of planes with a specified value.

Usage:

CALL INSERT (IPS, NPS, IP, NVALUE)

Parameters:

IPS - Stack of planes into which object will be inserted
NPS - Number of planes in IPS
IP - Index of plane containing object to be inserted
NVALUE - Value object will have when inserted into stack

Execution:

Each point of the stack IPS(1),...,IPS(NPS) that corresponds to a 1 in plane IP is set to the value specified by the NPS low order bits of NVALUE. Other points of the stack are not altered.

Integer Function KGB

Purpose:

To set an integer variable to the value of a specified bit following the first bit in a specified word.

Usage:

IB = KGB (LOCATN, N)

Parameters:

LOCATN - Address constant (generated by Integer Function LOC) pointing to a word

N - Position of the bit desired counting from left to right beginning at the first bit in the word pointed to by LOCATN and continuing to successive words if necessary

Execution:

IB is set to 0 or 1 according to the value of the bit specified by LOCATN and N.

Integer Function KGC

Purpose:

To determine a specified character in a Hollerith field.

Usage:

KAR = KGC(KV, N)

Parameters:

KV - Variable containing first characters in the Hollerith field

N - Position of the character desired counting from left to right, beginning at the first character in KV and continuing to successive words

Execution:

The bit configuration for the Hollerith character specified by KV and N is put in KAR, right adjusted with leading zeros.

Subroutine KPB

Purpose:

To set a specified bit to 0 or 1.

Usage:

CALL KPB (I, LOCATN, NBIT)

Parameters:

I - Integer which is 0 if bit is to be set to 0 or not 0 if the bit is to be set to 1

LOCATN - Address constant (generated by Integer Function LOC) pointing to a word

N - Position of the bit to be set, counting from left right at first bit in the word pointed to by LOCATN and continuing to successive words if necessary

Execution:

The bit specified by LOCATN and N is set to 0 if I = 0 or to 1 if I $\neq$ 0.

Subroutine KPC

Purpose:

To insert a character into a specified position in a Hollerith field.

Usage:

CALL KPC (KAR, KV, N)

Parameters:

KAR - Variable containing the character to be inserted, right adjusted
KV - Variable containing the first characters in the Hollerith field
N - Position into which the character is to be inserted counting from left to right, beginning at the first character in KV and continuing to successive words

Execution:

The Hollerith character in KAR is inserted into the position specified by KV and N.

Subroutine LABEL

Purpose:

To insert a picture of a given string of characters into a stack of planes with a given value.

Usage:

CALL LABEL (IPO, NPO, L, NL, KX, KY, NVALUE, MAG)

Parameters:

IPO	- Array of plane indices
NPO	- Number of planes in stack IPO
L	- Array containing characters (packed as many as possible per word)
NL	- Number of characters in L
KX, KY	- Column and row in which to begin inserting pictures of characters
NVALUE	- Value with which pictures of characters are inserted
MAG	- Parameter determining size of pictures of characters (they will be 5*MAG elements wide and 7*MAG elements high)

Execution:

Pictures of the NL characters in L are inserted into stack IP with value NVALUE. The characters in NL may be any which are valid on the computer being used. Each character picture is a subset of a 5 by 7 matrix of squares of size MAG. The character pictures are separated by MAG elements. NL characters will require a window of width 6*NL*MAG-MAG and height 7*MAG. Only the points which form the characters are altered. The background is not changed.

Programmer:

Edgar Butt

Integer Function LDL

Purpose:

To provide the locations of four common direction lists.

Usage:

L = LDL(K)

Parameters:

L - Integer variable that will be set to the location of the requested direction list
K - Integer indicating direction list. Possible values and corresponding direction lists are:

4 - four nearest neighbors
5 - four nearest neighbors and the point itself
8 - eight nearest neighbors
9 - eight nearest neighbors and the point itself

Execution:

The four direction lists are stored in this subroutine. K is examined and the appropriate address constant is returned. If K is not one of the values listed above, an error message is printed and execution is terminated.

Programmer:

Edgar Butt

Subroutine LETTER

Purpose:

To read characters from a card for use as print characters by Subroutine PRINT.

Usage:

CALL LETTER

Execution:

32 pairs of print characters are read from a card with a 32A2 format. See Subroutine PRINT for a discussion of print characters.

Subroutine LISTXY

Purpose:

To determine the points in a window of a plane which have the value 1.

Usage:

CALL LISTXY(IP, IW, JARRAY, J)

Parameters:

IP - Index of plane being examined
IW - Window
JARRAY - Array whose elements will be set to indicate the location of points with value 1. JARRAY must be dimensioned at no less than the value of J when the subroutine is called
J - Integer variable that is set by the user to indicate the maximum number of points to record in JARRAY and is set by the subroutine to indicate the number of points actually recorded

Execution:

The value of J is recorded, and then J is set to 0. Then successive rows of window IW are examined from left to right, beginning with the top row. When a point with value 1 is found, J is increased by 1. Then, if J does not exceed its initial value, the column and row of the point are packed into JARRAY(J) by Integer Function COORD. If J exceeds its initial value, however, the point is not recorded and execution of the subroutine is terminated. The user may use Subroutine UNCOOR to unpack the row and column of each point.

Integer Function LOC

Purpose:

To determine the machine address of a variable.

Usage:

L = LOC(M)

Execution:

L is set to the address of the variable M. This subroutine is frequently used to set a variable to the location of a direction list.

Subroutine LOCATE

Purpose:

To supervise the internal storage of planes and the movement of blocks of planes between internal and external storage.

Usage:

CALL LOCATE (NP, IP, IB, LOCATN)

Parameters:

NP	-	Number of plane indices in array IP
IP	-	Array of plane indices
IB	-	Block number
LOCATN	-	Array which will receive or already contains address constants

Labeled common and internal variables:

The array PLANE in the labeled common block /PIX/ contains a status word for each plane. PLANE(I) is zero if the I-th plane is not in use and is nonzero if it is in use.

The labeled common block /BUF/ contains the buffers which are used to store blocks of planes.

An activity queue of the buffers is maintained. The variable JOLD points to the head of the queue and JNEW points to the tail. If there are any unused buffers they are always at the head of the queue. JMT points to the last unused buffer. Active buffers are moved to the end of the queue.

The array BLOCK (dimensioned at 5, NBUF) in the labeled common block /BUX/ contains 5 control words for each buffer in /BUF/. BLOCK(I, J) is the I-th control word for the J-th buffe The function of each control word is:

1. Buffer status. A negative value means the buffer is unused. A zero means the buffer contains a block whic may be moved to external storage if necessary. A positive integer means the buffer contains a block which is being operated on and hence must remain in th buffer. The positive integer indicates the number of subroutines that are operating on the block.

2. Plane number of the block currently or most recently in the buffer.
3. Block number of the block currently or most recently in the buffer.
4. Number of the buffer which follows on the queue. Zero indicates that this buffer is last on the queue.
5. Address constant pointing to the beginning of the buffer.

Execution:

Initialization. On the first call to LOCATE only, the block data initialization is tested and buffer control words, plane status words and several internal variables are initialized.

Selecting a new plane. If IB is a valid block number, then for each I = 1,..., NP such that LOCATN(I)=0 and IPI(I) =0, an unused plane is selected and IP(I) is set to the negative of the plane number to inhibit input. The action described in the next paragraph is then taken.

Locating a block. If IB is a valid block number, then for each I = 1,..., NP such that LOCATN(I)=0, the buffers are searched for block IB of plane |IP(I)|. If no buffer contains the block, a buffer must be assigned to the block. If there are no unused buffers, the buffer nearest the head (JOLD) of the queue with a status word of 0 is made available by moving its contents to external storage via Subroutine DISK. If IP(I)>0, block IB of plane IP(I) is moved from external storage to the available buffer. If IP(I)<0, movement is inhibited. Subroutines which are going to assign new values to every point in a block would call LOCATE with negative plane numbers to inhibit unnecessary movement. Finally, the buffer which contains the block or has been assigned to the block is moved to the end (JNEW) of the queue, its status word is increased by 1 or set to 1, whichever is larger, and LOCATN(I) is set to the address of the buffer.

Releasing a block. If IB is a valid block number, then for each I = 1,..., NP such that LOCATN(I)≠0, the status word of the buffer containing block IB of plane |IP(I)| is reduced by 1 and LOCATN(I) is set to 0. This action is requested by a subroutine that has previously located a block and is now finished operating on it.

Releasing a plane. If IB = 0, then I = 1,..., NP, such that IP(I)≠0, plane |IP(I)| is returned to unused status, buffers containing blocks of plane |IP(I)| are returned to unused status, and IP(I) is set to 0. 'CALL LOCATE (NP, IP, 0, LOCATN)' is equivalent to 'CALL RELEAS (NP, IP)'.

Simplified LOCATE:

There is a simplified version of this subroutine which may be used when external storage of planes is not implemented. It requires that there be only one block per plane and that the number of planes equal the number of buffers. Using the simplified version will save storage space and execution time.

Error Messages:

When certain errors are encountered, this subroutine will print an error message, call Subroutine SNAP and stop execution. The error message is 'LOCATE ERROR N' where the values of N and their meanings are:

1. IB has an illegal value.
2. |IP(I)| exceeds the number of planes for some I.
3. An unused plane has been requested when all planes are in use.
4. A buffer is needed but all buffers have positive status words.
5. The block data program has not been loaded.
6. There has been a request for a block of a plane which is not in use.
7. The simplified version of Subroutine LOCATE has been loaded when the standard version is needed.

Subroutine MARK

Purpose:

To shift an input plane in each direction of a direction list and combine the results with logical ors on an output plane.

Usage:

CALL MARK(LDL, IPO, IPI, IPC)

Parameters:

LDL - Address constant pointing to a direction list IDL
IPO - Output plane index
IPI - Input plane index
IPC - Context plane index

Execution:

The input plane is shifted according to each direction in direction list IDL. The results are combined on an initially clear scratch plane with logical ors. If IPC = 0, plane IPO is set equal to the scratch plane. If IPC≠0, plane IPO is the logical and of plane IPC and the scratch plane.

Subroutines MAXIMA, MINIMA

Purpose:

To determine the points in a stack of planes which are maxima or minima with respect to a specified set of neighbors.

Usage:

```
CALL MAXIMA(LDL, IPO, IPI, NPI)
CALL MINIMA(LDL, IPO, IPI, NPI)
```

Parameters:

LDL - Address constant pointing to direction list IDL
IPO - Index of the output plane
IPI - Array of indices of input stack
NPI - Number of planes in the input stack

Execution:

Subroutine MAXIMA (MINIMA) sets each point of plane IPO to 1 if the corresponding point in the stack IPI(1),...,IPI(NPI) is not less than (not greater than) any of its IDL-neighbors. If a neighbor of a point is off the plane, it has the value of 1.

Subroutines NAND, NANDS, NOR, NORS

Purpose:

To set a plane (stack of planes) to the complement of the result of performing a logical "and" or a logical "or" on two planes (stacks of planes).

Usage:

```
CALL  NAND  (IPO, IPI1, IPI2)
CALL  NANDS (IPO, IPI1, IPI2, NP)
CALL  NOR   (IPO, IPI1, IPI2)
CALL  NORS  (IPO, IPI1, IPI2, NP)
```

Parameters:

IPO - Array of indices of output planes
IPI1 - Array of indices of input planes
IPI2 - Array of indices of input planes
NP - Number of indices in IPO, IPI1, and IPI2

Execution:

Points on planes IPO(I) are set according to the values of corresponding points in planes IPI1(I) and IPI2(I) for I = 1,..., NP according to the table below. For subroutines NAND and NOR, NP is assumed to be 1.

IPI1(I)	IPI2(I)	IPO(I)	
		NAND, NANDS	NOR, NORS
0	0	1	1
0	1	1	0
1	0	1	0
1	1	0	0

Programmer:

Edgar Butt

Subroutine NEIGHB

Purpose:

To list the values of the neighbors of a given point in a stack of planes.

Usage:

CALL NEIGHB(IP, NP, KX, KY, IDL, JARRAY)

Parameters:

IP	- Array of indices of the planes in the stack
NP	- Number of planes in stack IP
KX	- Column of the point
KY	- Row of the point
IDL	- Direction list which defines neighbors
JARRAY	- Integer array to receive values of the neighbors of the point

Execution:

READZ is called IDL(1) times to put the values of the IDL-neighbors of the point in column KX, row KY of the stack of planes IP(1),...,IP(NP) in JARRAY.

Subroutine NULL

Purpose:
To determine whether or not all the points on a plane are zero.

Usage:
CALL NULL (IP, JGT)

Parameters:
IP - Plane index
JGT - Parameter set to indicate result of test

Execution:
If all points of plane IP are zero, JGT is set to 1. Otherwise, JGT is set to 2.

Subroutine OUTPUT, OUTROW

Purpose:

To move the values from successive rows of a window of a stack of planes to an array.

Usage:

```
          CALL OUTPUT(NP,IW,IP,JARRAY,JGT)
10        .
          . (Instructions to process values for one row of the
          . window which have been put into JARRAY)
          .
          CALL OUTROW
          GO TO (10,20),JGT
20        .
          .
          .
```

Parameters:

NP	-	Number of indices in IP
IW	-	Window
IP	-	Array of plane indices
JARRY	-	Integer array large enough to hold values for one row of the window
JGT	-	Computed go to parameter set by the subroutine

Execution:

OUTPUT and OUTROW are entries to the same subroutine. When OUTPUT is called, the subroutine is initialized, JGT is set to 1, and the values from the first row of the window are moved to JARRAY. When OUTROW is called the values for the next row, if there is another row, are moved to JARRAY. If there is not another row in the window when OUTROW is called, JGT is set to 2.

Subroutine PF

Purpose:

To facilitate bit-plane decoding. Specifically, to set bits in a specified position in successive words of an array to the values of successive bits, beginning with a specified bit.

Usage:

CALL PF (LWORD, BIT, ARRAY, N, POS)

Parameters:

LWORD - Address constant pointing to the word containing the first bit to be examined
BIT - Number, counting from 1 on the left, of the first bit in the word pointed to by LWORD that will be examined
ARRAY - Array of N words in which bits will be set
N - Number of bits to be set
POS - Word containing a single 1-bit which specifies the bit to be set in each word of the array

Execution:

The bits in the position specified by POS in the words ARRAY(1), ..., ARRAY(N) are set to the values of the N consecutive bits beginning with the one specified by LWORD and BIT. It is assumed that when FP is called the bits in the position specified by POS in the words ARRAY(1),..., ARRAY(N) have the value 0. Thus action is required only if a bit is to be set to 1.

Integer Function POINT

Purpose:

To set an integer variable to a specified element in an array located by an address constant.

Usage:

```
INTEGER POINT
I = POINT (LOCATN, N)
```

Parameters:

LOCATN - Address constant (generated by Integer Function LOC) pointing to the first element of an array
N - Positive integer specifying an element of the array

Execution:

I is set to the value of the Nth element of the array pointed to by LOCATN.

Subroutine POLYGN

Purpose:

To create a convex polygonal region with a given value on a stack of planes.

Usage:

CALL POLYGN (IPO, NPO, NVALUE, NVERT, COORDS)

Parameters:

IPO - Array of indices of planes in output stack
NPO - Number of planes in stack IPO
NVALUE - Value to which points in polygonal region will be set
NVERT - Number of vertices in polygon
COORDS - Array containing the coordinates of the vertices in the polygon. The dimension of COORDS must be (2,NVERT). COORDS(1,N) and COORDS(2,N) are the column and the row of the N-th vertex. The first vertex must have a minimal row coordinate, that is, it must be one of the vertices nearest the top of the picture. Succeeding vertices must be given in clockwise order around the polygon. COORDS is a real array to allow vertices which do not coincide with picture elements.

Execution:

The points on stack IPO which are inside or on the boundary of the polygon defined by NVERT and COORDS are set to NVALUE. The points are set a row at a time, top row first, proceeding from left to right using subroutine WRITEP.

Programmer:

Edgar Butt

Note: The polygon need not be convex, provided that its intersection with each row of the stack is connected.

Subroutines PRINT, SPRINT

Purpose:

To print a window of a digitized picture.

Usage:

```
CALL PRINT(NP,IW,IP)
CALL SPRINT(NP,IW,IP,NSP)
```

Parameters:

- NP - Number (not greater than 5) of planes which contain the picture
- IW - Window of the picture to be printed
- IP - Array of indices of the planes which contain the picture
- NSP - Number of blank spaces to appear to the left of the printed image of the window

Execution:

The window of the digitized picture stored in IP(1),..., IP(NP) is displayed on the line printer as a rectangular array of print characters. The window is normally centered horizontally on the page, but the alternate entry, SPRINT, allows the programmer to specify horizontal spacing. This routine performs no vertical spacing. Printing begins wherever the printer is positioned when this subroutine is called. Windows which extend beyond the picture are truncated.

The print characters which correspond to the digitized values 0 to 31 are stored in a 32-word block of labelled common (/LTX/LT(32)). Each word contains two characters left adjusted. When the print routine is selecting a print character for the digital value I(I=0,...,31), the two characters in LT(I+1) are examined. If the second character is blank, the first character is used to represent I. If the second character is a prime ('), the first character will be used to represent I in odd columns, and a blank will be used to represent I in even columns. In all other cases, I will be represented by the first character overstruck with the second character.

The standard characters used to represent 0 to 31 are BLANK, 1,2,...,9,A,B,...V. See subroutine LETTER and EXCHAR for information on changing the print characters.

Subroutine PRINTS

Purpose:

To automatically print a window of a stack of planes that is wider than the printer.

Usage:

CALL PRINTS (NP, IW, IP)

Parameters:

NP - Number of planes in stack IP
IW - Window to be printed
IP - Array of indices of planes in the stack to be printed

Execution:

The window is divided into vertical strips which are not wider than a page. Before each strip is printed the printer is positioned at the top of a page.

Programmer:

Edgar Butt

Subroutine PROP

Purpose:

To call Subroutine MARK a given number of times with a given set of direction lists

Usage:

```
CALL PROP(LDL,NDL,IPO,IPI,IPC,NPROPS)
```

Parameters:

LDL	-	Array of address constants pointing to direction lists IDL
NDL	-	Number of direction lists pointed to by LDL
IPO	-	Output plane index
IPI	-	Input plane index
IPC	-	Context plane index
NPROPS	-	Number of times that MARK is called

Execution:

Plane IPI is copied on plane IPO. Then IPO is used as the input plane for NPROPS calls to MARK using the direction lists pointed to by LDL(1),..., LDL(NDL),LDL(1),... The output from each call to MARK is ored onto plane IPO before the next call to MARK. Plane IPC is used as the context plane in the calls to MARK.

Subroutine PTHS

Purpose:

To print a histogram

Usage:

CALL PTHS (IHIST, NHIST, I)

Parameters:

IHIST - Array containing frequencies to be used for the histogram
NHIST - Dimension of IHIST
I - Integer that will be used to label the first frequency in the histogram

Execution:

The maximum frequency in IHIST is used to determine a scaling factor so that the histogram will be as wide as a page. Then each frequency is printed followed by an appropriate number of asterisks. The first frequency is labelled I and successive frequencies are labelled with successive integers. Frequencies of 0 at the beginning and end of the histogram are not printed.

Subroutine GLHS generates an array of the frequencies of values in a window of a stack of planes.

Programmer:

Li-Chen Wang

Subroutine PUNCH

Purpose:
To output values from a window of a stack of planes to cards.

Usage:
CALL PUNCH (NP, IW, IP)

Parameters:
NP - Number of planes in stack IP (not more than 5)
IW - Window
IP - Array of plane indices

Execution:
One card is punched with IW(1). . . IW(4) and NP in 5I5 format followed by the word 'PUNCH'. Then characters corresponding to values for each row are punched with an 80A1 format. Card characters BLANK, 1, 2, ..., 9, A, B, ..., V correspond to values 0, 1, ..., 31. Finally, a card is punched with the words 'END PUNCH'.

Real Function RANDNO

Purpose:

To provide a sequence of pseudo-random real numbers between 0 and 1.

Usage:

X = RANDNO(NR)

Parameters:

X - Real variable that the subroutine will set to a number between 0 and 1
NR - Integer variable that will be altered by RANDNO

Execution:

RANDNO generates a periodic sequence of real numbers between 0 and 1. The period, however, is sufficiently large to make the numbers seem random. NR is used to store a positive integer which indicates which element in the sequence was last returned. When RANDNO is called, it examines NR. If NR is a positive integer, the next element of the sequence is generated and stored in X, and NR is modified. If NR is not positive, NR is set to a positive integer obtained by making a hash total of some active section of core and then execution proceeds as before.

The user may force the starting point in the sequence by setting NR to a positive integer before calling RANDNO the first time.

Subroutine RANPIC

Purpose:

To add noise (0's or 1's) to a window of a plane with a given probability.

Usage:

CALL RANPIC (IPO, IW, PROB, NVALUE, NR)

Parameters:

IPO	-	Plane index
IW	-	Window
PROB	-	Probability with which noise is to be generated
NVALUE	-	Value of noise (0 or 1)
NR	-	Integer variable used by Integer Function RANDNO

Execution:

If IPO=0, plane IPO is cleared. Then points in window IW of plane IPO are set to 0 (if NVALUE=0) or 1 (if NVALUE≠0) with probability PROB. See Integer Function RANDNO for a discussion of NR.

Subroutines READP, RP, ENDRP

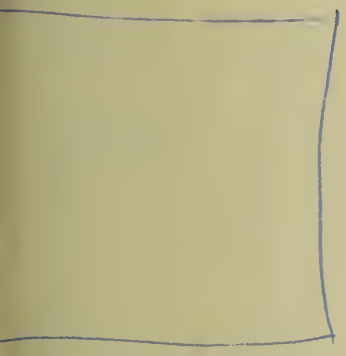
Purpose:

lues of points in a stack of

PO, KX, KY, JVALUE)

SET KX, KY)

) USE JVALUE)



: plane indices
of planes in stack IPO
and row of a point whose value will
rmined
e to receive the value of one point

nitialization entry is called, the subrou-
ies will be tested and linkage to
s established. Each time RP is
t to the value of the point de-
ent values of KX and KY. ENDRP
the user has finished reading

: is more efficient than subroutine
ny points because no argument link-
entry WP and because fewer calls to
subroutine [illegible] are made.

Programmer:

Edgar Butt

RANPIC

Subroutine RANPIC

Purpose:

To add noise (0's or 1's) to a wind
given probability.

Usage:

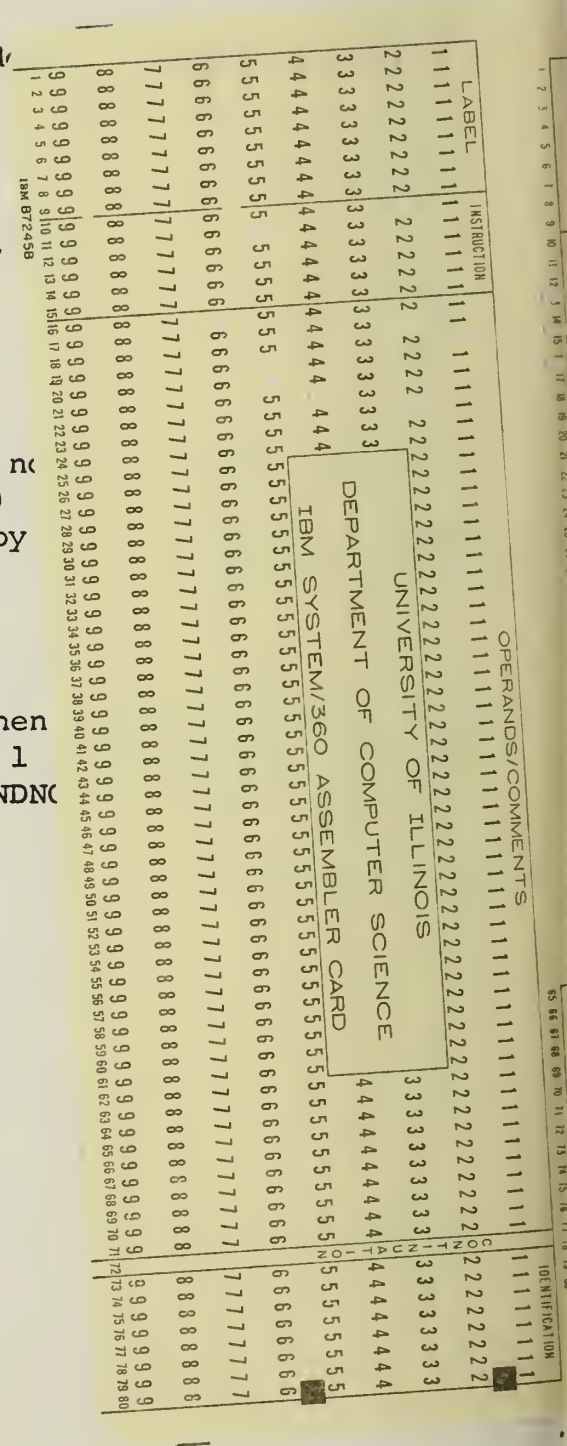
CALL RANPIC (IPO, IW, PROB, NVALUE,

Parameters:

IPO	-	Plane index
IW	-	Window
PROB	-	Probability with which n
NVALUE	-	Value of noise (0 or 1)
NR	-	Integer variable used by

Execution:

If IPO=0, plane IPO is cleared. Then
plane IPO are set to 0 (if NVALUE=0) or 1
bability PROB. See Integer Function RANDN
NR.



Subroutines READP, RP, ENDRP

Purpose:

To determine the values of points in a stack of planes.

Usage:

```
     CALL READP (IPO, NPO, KX, KY, JVALUE)
     DO 10 I = 1, N
     .
     . (INSTRUCTIONS TO SET KX, KY)
     .
     CALL RP
     .
     . (INSTRUCTIONS TO USE JVALUE)
  10 .
     CALL END RP
```

Parameters

IPO - Array of plane indices
NPO - Number of planes in stack IPO
KX, KY - Column and row of a point whose value will be determined
JVALUE - Variable to receive the value of one point

Execution:

When READP, an initialization entry is called, the subroutine notes which planes will be tested and linkage to KX, KY, and JVALUE is established. Each time RP is called, JVALUE is set to the value of the point determined by the current values of KX and KY. ENDRP must be called when the user has finished reading values.

This subroutine is more efficient than subroutine READZ for reading many points because no argument linkage is required for entry WP and because fewer calls to subroutine LOCATE are made.

Programmer:

Edgar Butt

Subroutine READZ

Purpose:

To determine the value of a point in a stack of planes.

Usage:

```
CALL READZ(JVALUE,KX,KY,NP,IP)
```

Parameters:

JVALUE	-	Variable which will be set to the value of the point
KX	-	Column of the point
KY	-	Row of the point
NP	-	Number of indices in IP
IP	-	Array of plane indices

Execution:

JVALUE is set to the value of the point in column KX, row KY of the stack of planes IP(1), ..., IP(NP).

Subroutine RELEAS

Purpose:
To return a plane currently being used to unused status.

Usage:
CALL RELEAS (NP, IP)

Parameters:
NP - Number of planes to be released
IP - Array of indices of planes to be released

Execution:
Planes IP(1),..., IP(NP) are released and the indices IP(1),..., IP(NP) are set to zero.

Subroutine ROTATE

PURPOSE: To perform rotations and dilatations on a stack of planes

USAGE: CALL ROTATE(IPO, NP, IWO, IPI, CXI, CYI, THETA, ALPHA, ISW)

PARAMETERS:

- IPO - array of indices of planes in the output stack
- NP - number of planes in each of the stacks IPO and IPI
- IWO - window of output stack to be filled
- IPI - array of indices of planes in the input stack
- CXI, CYI - column and row of point on input stack which will be the center of rotation and dilatation. CXI and CYI are real to allow a center which is not a grid point
- THETA - angle of rotation in radians
- ALPHA - dilatation factor. ALPHA must be positive. It it is less than 1, contraction will occur. If it is equal to 1 there will be no dilatation. If it is greater than 1, expansion will occur
- ISW - integer which is not 0 if linear interpolation is desired or 0 if it is not desired

EXECUTION: Subroutine INPUT is used to fill the output window with the result of performing the desired transformation on the input picture. The center of rotation and dilatation on the input picture is mapped to the center of the output window. To determine the value of a point, P, in the output window the reverse transformation is applied to P. The resulting point, Q, is not necessarily a grid point of

the input plane. If ISW = 0 the value of the grid point nearest Q is used as the output value. If ISW $\neq$ 0, the output value is determined by linear interpolation of the values of the four grid points surrounding Q. Subroutine READP is used to read values on the input stack.

E. Butt
February, 1969

Subroutine RPROP

Purpose:

To generate a plane which has a rectangle of 1's of given length and width for each 1 on an input plane. Each 1 on the input plane will correspond to a corner of a rectangle on the output plane. The sides of the rectangles are parallel to the edges of the plane.

Usage:

CALL RPROP (IPO, IPI, IPC, NX, NY)

Parameters:

- IPO - Index of output plane
- IPI - Index of input plane
- IPC - Index of context plane
- NX, NY - Shift parameters which determine the size and orientation of the rectangles

Execution:

Each point in plane IPO is set to 1 if its column number is from KX to KX+NX and its row number is from KY-NY to KY where KX and KY are the column and row of any point in plane IPI with value 1.

If IPC is not 0, then plane IPO may differ from plane IPI only on points corresponding to a 1 in plane IPC.

Subroutine SEGMNT

Purpose:

To join two points on a stack of planes with a line segment of given value and width.

Usage:

```
CALL SEGMNT (IPO, NPO, NVALUE, NWIDTH, KX1, KY1,
             KX2, KY2)
```

Parameters:

IPO	- Array of indices of output stack
NPO	- Number of planes in stack IPO
NVALUE	- Value with which segment will be inserted
NWIDTH	- Width of line segment in picture elements
KX1, KY1	- Column and row of one endpoint
KX2, KY2	- Column and row of second endpoint

Execution:

The coordinates of the endpoints and the width are used to calculate the coordinates of a rectangle corresponding to the line segment. Subroutine POLYGN is then called to generate the rectangle.

Programmer:

Edgar Butt

~~NOTE: |KX1 - KX2| + |KY1 - KY2| > 0~~
~~else error (divide by 0)~~ fixed

Subroutines SETOP, OP

Purpose:

To perform the word and bit manipulation necessary for logical or bit counting operations on planes.

Usage:

```
CALL  SETOP (OPCODE, LENGTH)
CALL  OP (LOCATN)
```

Parameters:

LOCATN - Array of 1, 2, or 3 address constants pointing to arrays designated ARRAY1, ARRAY2, and ARRAY3

OPCODE - Integer whose values and meanings are:

- 0 - Perform a null test on ARRAY1
- 1 - Set all bits in ARRAY1 to 0
- 2 - Set all bits in ARRAY1 to 1
- 3 - Copy ARRAY2 into ARRAY1
- 4 - Copy the complement of ARRAY2 into ARRAY1
- 5 - Set ARRAY1 to the logical and of ARRAY2 and ARRAY3
- 6 - Set ARRAY1 to the logical and of ARRAY2 and the complement of ARRAY3
- 7 - Set ARRAY1 to the logical or of ARRAY2 and ARRAY3
- 8 - Set ARRAY1 to the logical or of ARRAY2 and the complement of ARRAY3
- 9 - Set ARRAY1 to the logical exclusive or of ARRAY2 and ARRAY3
- 10 - Set ARRAY1 to the logical exclusive or of ARRAY2 and the complement of ARRAY3
- 11 - Count the number of 1-bits in ARRAY1
- 12 - Count the number of 0-bits in ARRAY2

LENGTH - Length in machine addresses of arrays pointed to by LOCATN

Execution:

SETOP is an initialization entry to the subroutine. When OP is called the operation specified in the last call to SETOP is performed on the arrays pointed to by LOCATN. The lengths of the arrays are also determined by the previous call to SETOP. The arrays need not be distinct.

The null test (OP = 0) and area operations are functions. For the null test a value of 0 is returned if all bits in ARRAY1 are 0; otherwise, a 1 is returned. For the area operations the number of 1-bits or 0-bits is returned.

Subroutines SETSH, SH

Purpose:

To perform the word and bit manipulation necessary for shifting the bits in consecutive words to the right or left.

Usage:

```
CALL  SETSH (COUNT, LENGTH)
CALL  SH (LOCIN, LOCOUT)
```

Parameters:

COUNT - Integer specifying number of positions each bit is to be shifted. Positive values are for right shifts, negative values are for left
LENGTH - Length in machine addresses of input and output arrays
LOCIN - Address constant pointing to input array
LOCOUT - Address constant pointing to output array

Execution:

SETSH is an initialization entry to the subroutine. When SH is called, the output array is set to the result of shifting the input array according to the count specified on the previous call to SETSH. Bits shifted beyond the output array are lost. Zeros are put in each bit position of the output array which received no bit from the input array. The input array is not altered. The input and output arrays must be distinct.

Subroutines SHIFT, SHIFTS

Purpose:

To shift a plane (stack of planes) onto itself or onto another plane (stack of planes).

Usage:

```
CALL SHIFT (IPO, IPI, NX, NY)
CALL SHIFTS (IPO, IPI, NX, NY, NP)
```

Parameters:

IPO - Array of indices of output planes
IPI - Array of indices of input planes
NX - Horizontal shift parameter
NY - Vertical shift parameter
NP - Number of indices in IPO and IPI

Execution:

Plane IPO(I) is set to the result of shifting plane IPI(I) NX elements horizontally (right is positive) and NY elements vertically (up is positive) for I = 1,2 ..., NP. For Subroutine SHIFT, NP is assumed to be 1.

Subroutine SLICE

Purpose:

To determine the points in a stack of planes whose values satisfy a relation with a given integer.

Usage:

```
CALL SLICE(IPO, IPI, NPI, KREL, NUM)
```

Parameters:

IPO - Index of output plane
IPI - Array of indices of planes in the input stack
NPI - Number of planes in the input stack
KREL - Integer specifying a relation. Possible values and their corresponding relations are:
1 - less than
2 - less than or equal to
3 - equal to
4 - greater than or equal to
5 - greater than
NUM - Integer with which points in the input stack are compared

Execution:

Each point of plane IPO is set to 1 if it corresponds to a point in stack IPI(1),...,IPI(NPI) whose value is in the relation specified by KREL to NUM. Other points in plane IPO are set to 0.

Subroutine SNAP

Purpose:

To provide the PAX user and the PAX subroutines with a means of obtaining memory dumps.

Usage:

```
CALL  SNAP
```

Parameters:

None

Execution:

SNAP essentially initiates a call to SYSERR. This provides a subroutine trace useful in debugging. If a dump is desired a user may so specify by including the standard JCL.

(i.e. add `//GO.SYSUDUMP  DD  SYSOUT = A` before the `//GO.SYSIN  DD *` card)

Subroutine SUMONE

Purpose:

To add (subtract) a plane algebraically to (from) a stack of planes and an associated sign plane.

Usage:

CALL SUMONE (IP, NP, IPS, IPI, IADSB)

Parameters:

IP	- Stack of planes for absolute value of the sum
NP	- Number of planes in IP
IPS	- Sign plane associated with IP
IPI	- Plane to be added or subtracted
IADSB	- Integer whose value is 1 for addition or 2 for subtraction

Execution:

When the subroutine is called, plane IPS has the value 1 at each point corresponding to points in stack IP that are to be considered negative. Negative zeros are not valid. When the subroutine is finished, stack IP and plane IPS have been altered to reflect the result of adding or subtracting plane IPI. Since overflow is not detected, the user must see that stack IP is sufficiently large.

Subroutine TMARK

Purpose:

To determine each point in a plane whose number of neighbors with the value 1 satisfies a relation with a given integer.

Usage:

CALL TMARK(LDL, IPO, IPI, IPC, KREL, NUM)

Parameters:

LDL - Address constant pointing to direction list IDL which defines neighborhoods
IPO - Index of output plane
IPI - Index of input plane
IPC - Index of context plane
KREL - Integer specifying a relation. Possible values and their corresponding relations are:
1 - less than
2 - less than or equal to
3 - equal to
4 - greater than or equal to
5 - greater than
NUM - Integer for comparison

Execution:

A point in plane IPO is set to 1 if the number of IDL-neighbors of the corresponding point in IPI is in the relation specified by KREL to NUM. Then, if IPC is not 0, plane IPC is anded onto plane IPO.

The direction list IDL may contain up to 127 directions. Directions in excess of 127 will be disregarded.

Subroutine TSHIFT

Purpose:

To determine whether a specified shift would shift any points with value 1 off a plane.

Execution:

CALL TSHIFT(IP, NX, NY, JGT)

Parameters:

IP - Plane index
NX - Horizontal shift count
NY - Vertical shift count
JGT - Computed go to parameter set by subroutine

Execution:

If the instruction, CALL SHIFT(IP, IP, NX, NY), would shift a point with value 1 off plane IP, JGT is set to 2. Otherwise, JGT is set to 1.

Subroutine UNCOOR

Purpose:

To retrieve two positive integers that were packed into an integer variable by Integer Function COORD.

Usage:

CALL UNCOOR(K, I, J)

Parameters:

K - Variable containing two positive integers
I - Variable to be set to the first integer
J - Variable to be set to the second integer

Execution:

I and J are set to the positive integers which were packed into K by Integer Function COORD.

Subroutine WINDOW

Purpose:

To clear a plane and then set all the points in a window to one.

Usage:

```
CALL WINDOW (IPO, IW)
```

Parameters:

IPO - Plane index
IW - Window

Execution:

Plane IPO is cleared and then all points in window IW are set to one. If the window is larger than the plane, results are not reliable.

Subroutine WRITEC

Purpose:
To set a point on a plane to 1.

Usage:
CALL WRITEC (IP, KX, KY)

Parameters:
IP - Plane index
KX - Column of the specified point
KY - Row of the specified point

Execution:
The point in the KX column, KY row of plane IP is set to 1.

Subroutines WRITEP, WP, ENDWP

Purpose:

To set the values of points in a stack of planes.

Usage:

```
     CALL WRITEP (IPO, NPO, KX, KY, NVALUE)
     DO 10 I=1, N
     .
     . (INSTRUCTIONS TO SET KX, KY, NVALUE)
     .
  10 CALL WP
     CALL ENDWP
```

Parameters:

IPO - Array of plane indices
NPO - Number of planes in stack IPO
KX, KY - Column and row of a point whose value will be set
NVALUE - Value a point will receive

Execution:

When WRITEP, an initialization entry, is called, the subroutine notes which planes are to be altered and linkage to KX, KY, and NVALUE is established. Each time WP is called, the value of the point determined by the current values of KX and KY is set to the current value of NVALUE. ENDWP must be called when the user is finished altering points.

This subroutine is more efficient than subroutine WRITEZ for writing many points because no argument linkage is required for entry WP and because fewer calls to subroutine LOCATE are made.

Programmer:

Edgar Butt

Subroutine WRITEZ

Purpose:

To set a point in a stack of planes to a given value.

Usage:

```
CALL WRITEZ(NVALUE, KX, KY, NP, IP)
```

Parameters:

NVALUE	-	Value to which point will be set
KX	-	Column of point
KY	-	Row of point
NP	-	Number of indices in IP
IP	-	Array of plane indices

Execution:

The point in the KX column, KY row of the stack of planes IP(1), ..., IP(NP) to set to NVALUE. Only the NP low order bits of NVALUE are considered.

OCT 15 1973

FEB 7- 1974

BOUND BY

UNIVERSITY OF ILLINOIS-URBANA
510.84 IL6R v.3 C002 v.308-315(1969)
TRANQUIL : a language for an array proca
3 0112 088404311